

README and Design for Grid Memory

This design for a grid memory using 2 BRAMs implements a 64x32 grid of 8x8 pixel cells. Each cell holds a 16-bit value (which could distinguish 65,536 objects, but this design only uses 2 bits/cell for 4 objects). To use this either MicroBlaze or a FSM would write the value of an object to the grid cell location (row/column) in the BRAM. When Scopeface reads the BRAM, it then draws the proper 8x8 pixel object on the screen at the cell location. The colors and shape of each type of object are defined in scopeface.

The BRAMs each have 1024 16-bit memory locations, so with 2 BRAMs there are 2048 locations. If we use 6 bits to specify the cell column and 5 bits to specify the cell row, then $2^6 \times 2^5 = 64 \times 32 = 2048$ cells. [see design on the following pages]

This design is modified from Lab2 and uses video/scopeface to read from the grid memory to display on the monitor through the HDMI port.

This Scopeface's ch1 is 16-bits wide which defines the object in each cell. This example only uses the 2 least significant bits defined as:

- 00 scopeface background art
- 01 Object#1
- 10 Object#2
- 11 Blue

If you want Scopeface to always draw an unchanging static image, it can use 00 to ignore the objects and draw the scopeface.

The bit file is included. If you'd like to build the project from the included files, you will need to recreate clock_wiz_0 similar to lab1.

The hardware FSM is set up to run a test image:

- Let exSel = '0' in graphics.vhd to run the test.

- The FSM has a delay (using a counter) between writes, so this can write much faster if you decrease the delay count.

To Use with MicroBlaze:

- Set up slv_regs to write to exSel, exRow, exCol, exData, and exWen

- Initialize exSel <= '1'

- For each cell to write to, let

- exRow <= the cell row to write to

- exCol <= the cell column to write to

- exData <= the object to draw in the cell

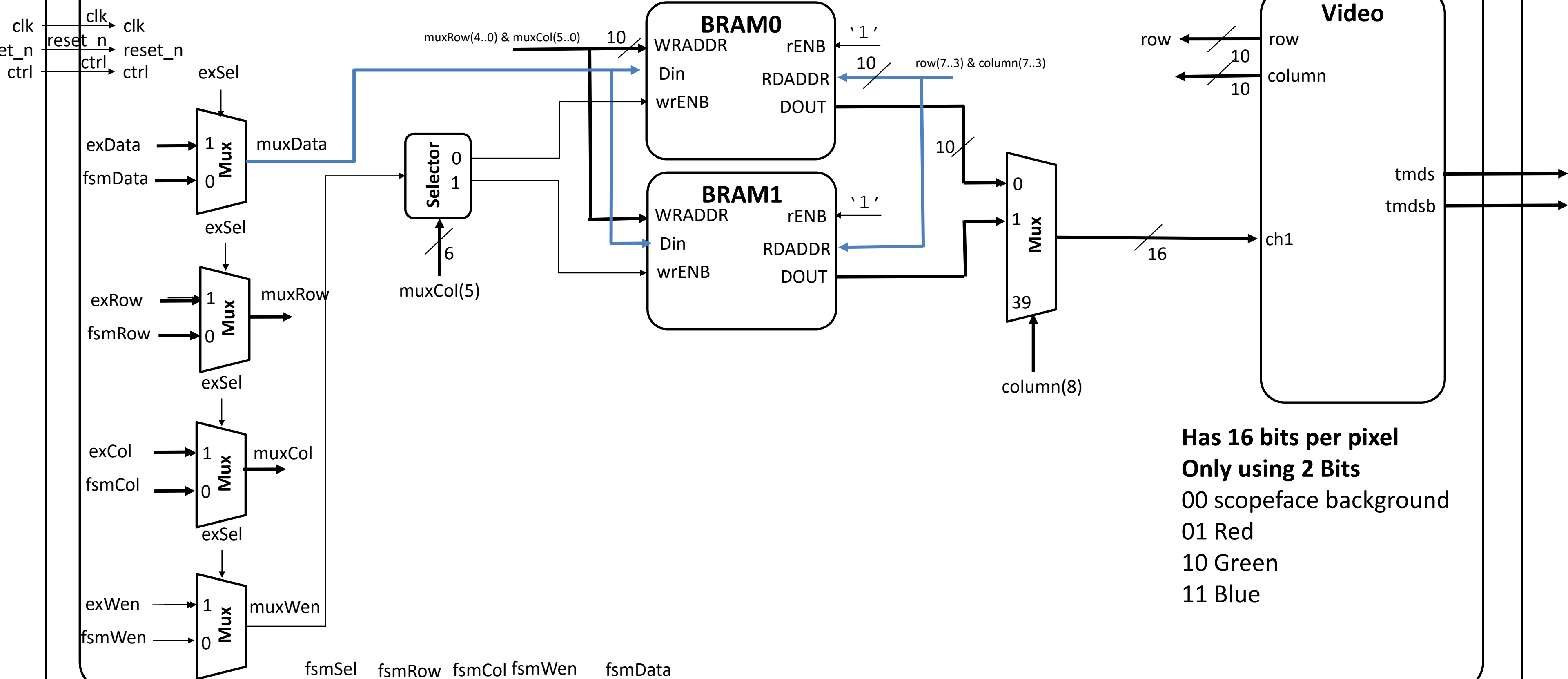
- exWen <= '1', to write

- exWen <= '0' to stop writing

- repeat for the next cell

graphics

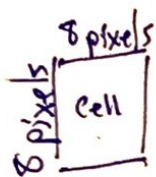
graphics_dp



Has 16 bits per pixel
Only using 2 Bits
00 scopeface background
01 Red
10 Green
11 Blue

graphics_fsm

clk
reset_n



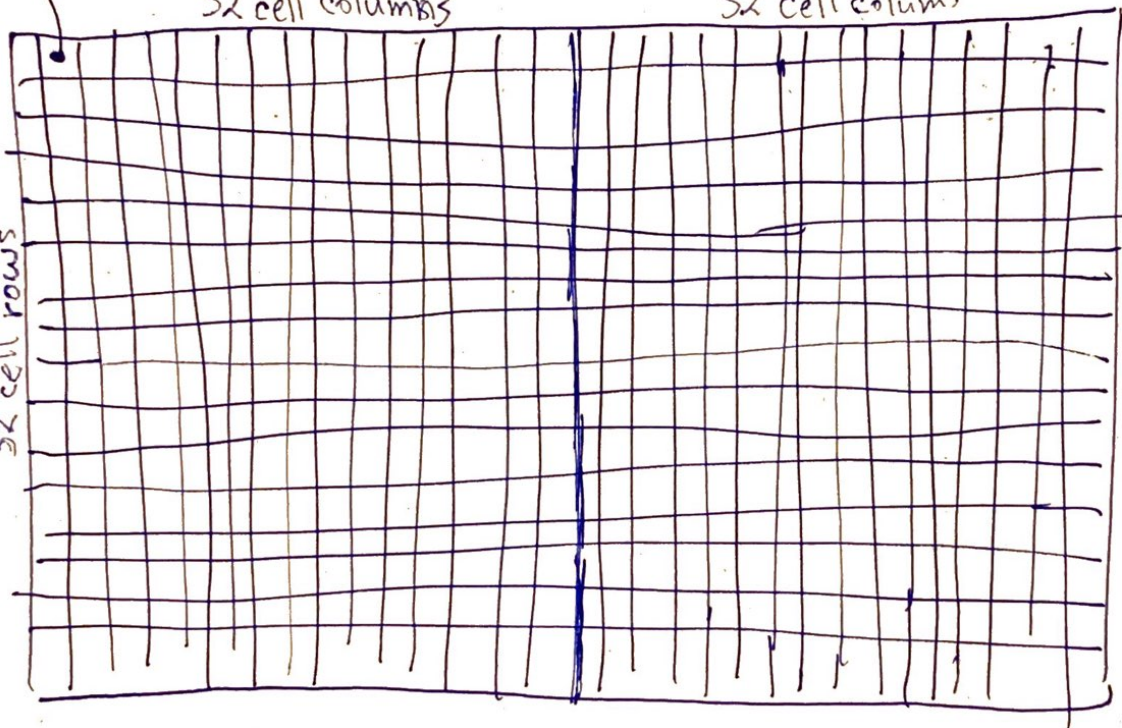
BRAMO

BRAMI

32 cell columns

32 cell columns

32 cell rows



64 cell columns → 6-bits

32 cell rows → 5-bits

	Scopeface		Pixel
	Extra	Cell	
row	2-bits	5-bits	3 bits
	Extra	Cell	Pixel
col	1-bit	6-bits	3 bits